Projeto de Front-End

Aula 2

Eduardo Mangeli

Agosto de 2023

Agenda

1. Git - o básico
   - Sistemas de Controle de Versão
   - Controle do Estado dos Arquivos
   - Característica do Git
   - Configurando o Git
   - Fundamentos e comandos básicos
   - Ignorando arquivos
2. Exercícios

Git - o básico

Git - o básico

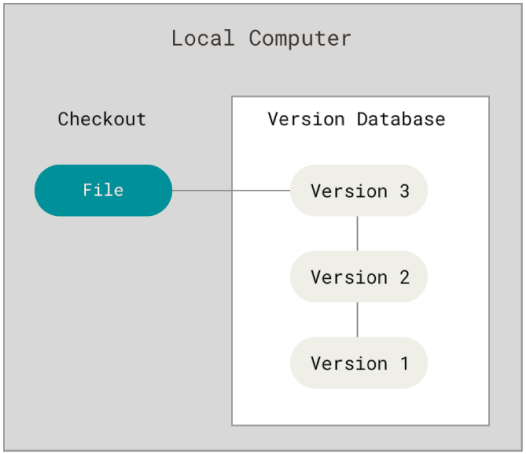
Sistemas de Controle de Versão

O que é?

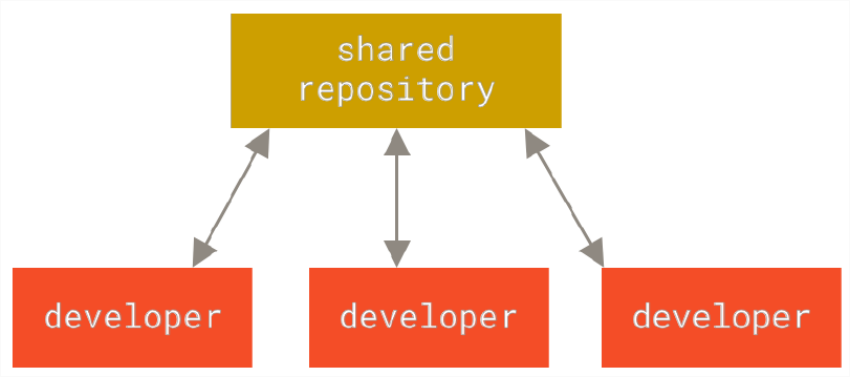
Sistemas de controle de versão é um sistema que registra alterações em um arquivo ou conjunto de arquivos ao longo do tempo para que você possa lembrar versões específicas mais tarde.

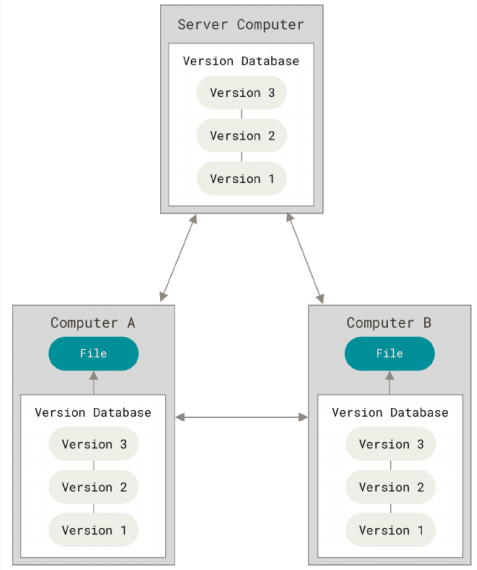
Sistemas Locais

- organizando sem sistema (manual)
- organizando com banco de dados de versões



Sistemas Centralizados

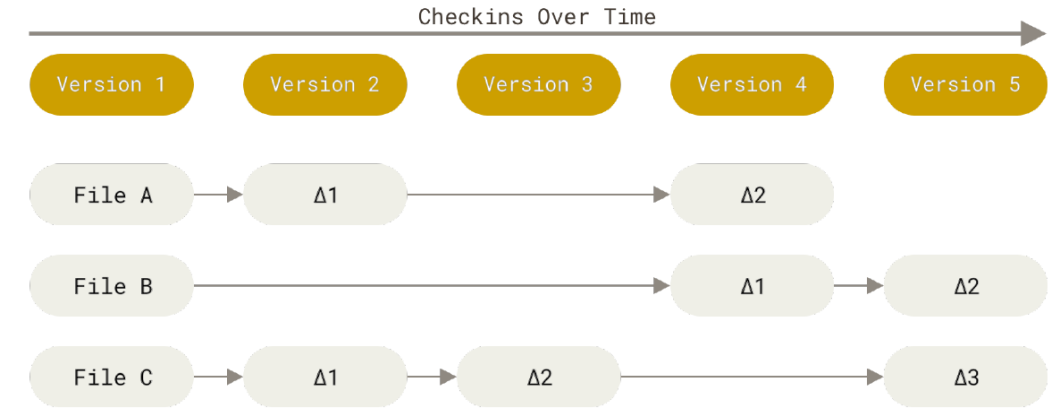



Server Computer
Version Database
Version 3
Version 2
Version 1
Computer A
File
Version Database
Version 3
Version 2
Version 1
Computer B
File
Version Database
Version 3
Version 2
Version 1

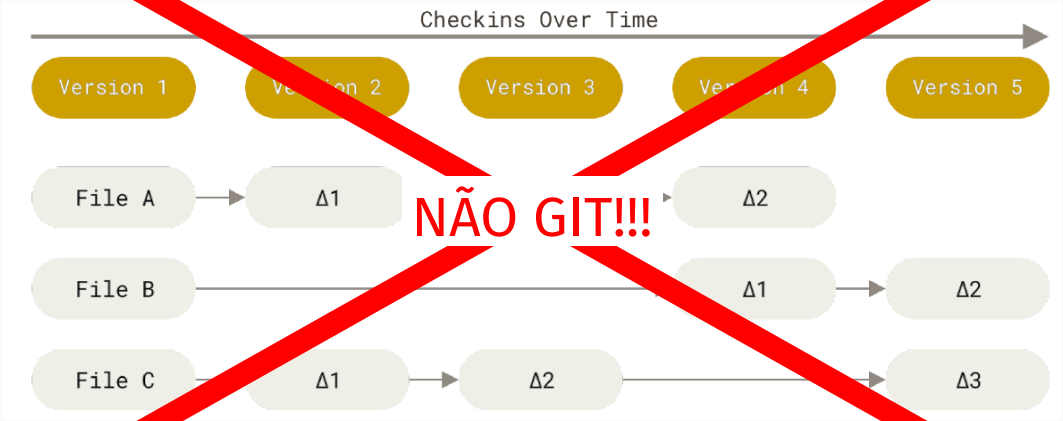
Git - o básico

Controle do Estado dos Arquivos

Diferenças

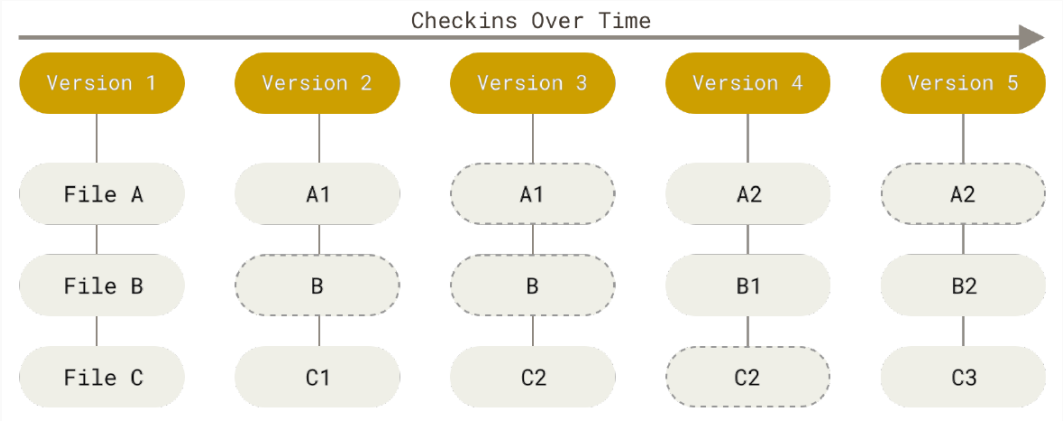

Checkins Over Time
Version 1
Version 2
Version 3
Version 4
Version 5
File A
Δ1
Δ2
File B
Δ1
Δ2
File C
Δ1
Δ2
Δ3

Diferenças



NÃO GIT!!!

Fluxo de Estados



Git - o básico

Característica do Git

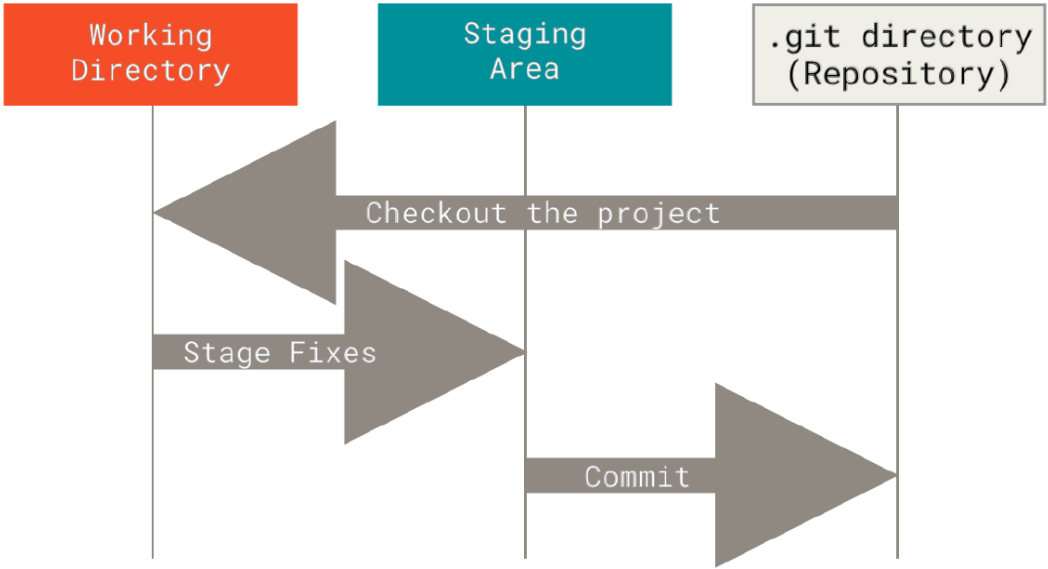
Alguns diferenciais do Git

- Mais operações locais
  - sem latência
  - independência
- Integridade dos arquivos – soma de verificação (checksum)
- Geralmente apenas adiciona dados

Os três Estados

- versionado (committed)
- modificado (modified)
- preparado (staged)

Três seções do projeto



Fluxo básico de trabalho

- modificar arquivos no diretório de trabalho (Working Dir)
- preparar arquivos (adicionando imagens a Staging Area)
- fazer commit

Git - o básico

Configurando o Git

git config

É o comando que configura variáveis necessárias ao funcionamento do git e que podem ser armazenadas em três lugares diferentes:

- `/etc/gitconfig`: Todo o sistema – `--system`
- `~/.gitconfig` ou `~/.config/git/config`: Nível de usuário. `--global`
- `.git/config`: específico para o repositório.

git config

É o comando que configura variáveis necessárias ao funcionamento do git e que podem ser armazenadas em três lugares diferentes:

- `/etc/gitconfig`: Todo o sistema – `--system`
- `~/.gitconfig` ou `~/.config/git/config`: Nível de usuário. `--global`
- `.git/config`: específico para o repositório.

Se estiver configurado em todos os lugares, qual vai funcionar?

Configurando sua identidade

```
$ git config --global user.name "Fulano de Tal"
$ git config --global user.email fulanodetal@exemplo.br
```

Verificando a configuração

```
$ git config --list
```

ou

```
$ git config <chave de configuração>
```

Git - o básico

Fundamentos e comandos básicos

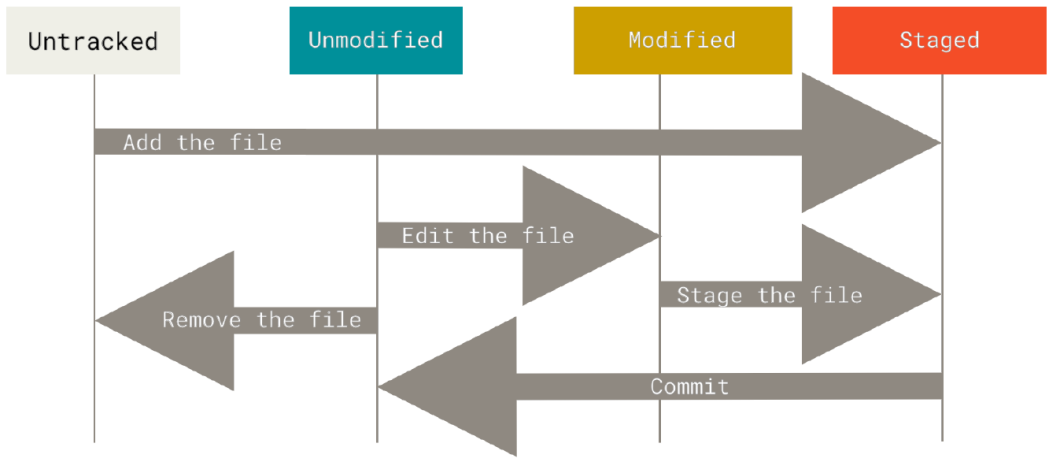
Como começar um repositório

Iniciando um repositório com o comando `git init`

OU

Clonando um repositório existente com `git clone <endereço>`

Status dos arquivos



Manipulando o status dos arquivos

- `git status` verifica o estado dos arquivos
- `git add` inclui um arquivo no sistema de versionamento ou uma imagem na Stage Area
- `git commit` registra as modificações da Stage Area no repositório
- `git rm` exclui um arquivo do sistema de versionamento (precisa fazer um `git commit` depois)

Outros comandos úteis

- `git log` ver os commits
- `git push` atualiza o repositório remoto com as alterações feitas localmente
- `git pull` atualiza o repositório local com as alterações do repositório remoto
- `git checkout <ramo>` muda o ramo de desenvolvimento
- `git tag` lista e cria etiquetas

Git - o básico

Ignorando arquivos

Motivação

Existem arquivos que geralmente não queremos incluir no sistema de versionamento, como:

- arquivos intermediários dos sistemas de compilação
- arquivos de configuração de IDEs
- arquivos resultados da compilação (geralmente estamos interessados no versionamento do código)

Para lidar com isso o git usa o arquivo `.gitignore` que contém regras de quais arquivos devem ser ignorados.

Regras

- Linhas em branco ou começando com # são ignoradas
- Os padrões que normalmente são usados para nomes de arquivos funcionam
- Você pode iniciar padrões com uma barra (/) para evitar recursividade
- Você pode terminar padrões com uma barra (/) para especificar um diretório
- Você pode negar um padrão ao fazê-lo iniciar com um ponto de exclamação (!)

Ferramentas

O github mantém uma boa lista de arquivos `.gitinore`
`https://github.com/github/gitignore`

O site `https://gitignore.io` constrói arquivos `.gitignore` personalizados

Sempre leia as mensagens dos programas!!!

Exercícios

1. Visite a página do repositório do github que você criou, leia o arquivo README (se houver) e clone seu repositório
2. Crie um novo arquivo de texto no diretório do seu repositório, adicione-o ao sistema de versionamento, faça o commit desse arquivo e o envie para o seu repositório remoto
3. Modifique o arquivo que você enviou para o repositório remoto usando o editor do próprio github (modifique o arquivo na internet, no seu repositório remoto)
4. Agora modifique o arquivo (uma outra modificação), no seu diretório local (no seu computador), adicione o arquivo modificado na StageArea, faça o commit e tente enviar para o repositório remoto
   4.1 Se não conseguir, tente entender o que aconteceu e resolver o problema.
5. Inclua um arquivo de áudio, com a terminação mp3, no seu repositório local
6. Faça um arquivo `.gitignore` que ignore qualquer arquivo mp3
7. Usa o commando `$ git add .` para incluir na Stage Area todos os arquivos modificados
8. Envie as modificações para o repositório remoto e verifique se o arquivo mp3 está lá (não deveria estar)